ALICE Analysis Framework

Andrei Gheata[*][†]
ISS/CERN
E-mail: Andrei.Gheata@cern.ch

The paper describes the current status of the offline analysis framework used in ALICE. The software was designed and optimized to take advantage of distributed computing resources and be compatible with ALICE computing model. The framework's main features: possibility to use parallelism in PROOF or GRID environments, transparency of the computing infrastructure and data model, scalability and data access performance. The framework provides a common "language" for all ALICE analysis users and is being heavily tested in view of the data to arrive soon.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7, 2008
Erice, Italy

[*]Speaker.
[†]On behalf of ALICE Offline Group

1. Introduction

Analysis procedures for experimental data are hard to formalize when the observables extend over a very large set of parameters. Among the other experiments at LHC, ALICE has a particularly complex detector system which will look into various observables in case of p-p, p-A and A-A interactions. Moreover, ALICE will be an unprecedented data producer for HEP [Figure 1] with data acquisition rates up to 1.25 GBytes/second, storing events in the range of 1 PByte each year.

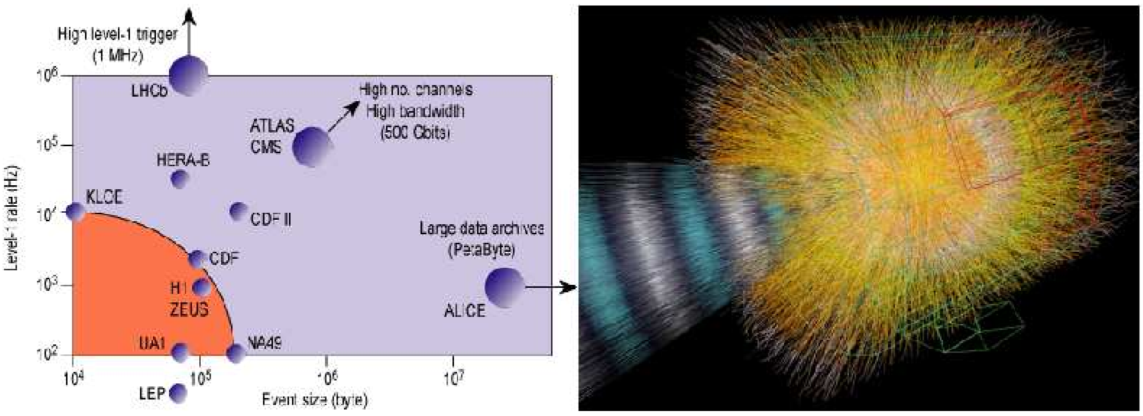


Figure 1: ALICE experiment will push the limits in terms of data storage and events size.

In this context, the ALICE core offline team developed a framework able to abstract data access patterns and develop a common analysis language for all experiment users. In a collaboration of about 1000 people, coordinating data analysis efforts is mandatory for insuring data availability for everyone and reproducibility of results. The set of tools developed for this purpose makes what we now call *analysis framework* and is currently being extensively used in the context of the analysis of data produced within ALICE physics data challenges.

2. Computing model and offline data flow

As the other LHC experiments, ALICE is using a multi-tier computing model that has to optimize and balance the load on unevenly-distributed computing resources. About half of these resources are concentrated into the CERN Tier0 and will have to store the raw data coming from the detector, do the first pass reconstruction, store calibration and event summary data.

A PROOF cluster of about 150 CPU-s will process data aiming for prompt reconstruction, calibration and preliminary data analysis while in the Tier0 disk buffer. Raw data and first pass event summary data (ESD) will be migrated also to the Tier1 centers, where subsequent reconstruction passes will be performed along with scheduled analysis. Both ESD-s and analysis object data (AOD) will be replicated here. The Tier2 centers are performing mostly simulation and user analysis, keeping also disk replicas of ESD-s and AOD-s.

Event summary data files are produced by reconstruction algorithms and contain information about: run and event numbers, trigger word, primary vertex, arrays of tracks and vertices and detector-specific information. The size of ESD events can be up to 15 MB for Pb-Pb and about 100 times smaller for p-p. The first step before actual analysis is preparing smaller objects based on ESD information, filtering out tracks and vertices which are unsuited for analysis together with some redundant detector info. This results in analysis object data (AOD) of a size 8 to 10 times smaller than ESD. AOD-s are intended to be directly used by user analysis or extended with custom information after pre-processing.

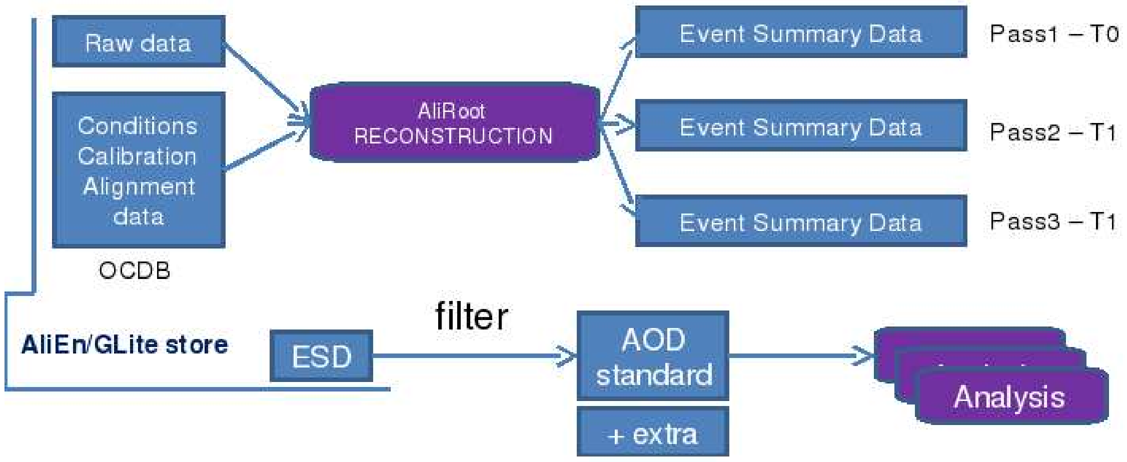


Figure 2: Offline data flow

3. Analysis framework

3.1 Analysis models and framework objectives

Three main data analysis models are foreseen:

- Prompt data processing for calibration, alignment, reconstruction and analysis run at CERN with PROOF. This analysis mode will be used to get a first look at the data and needs to have a very fast feedback. A prototype is currently being tested at ALICE CAF (CERN Analysis Facility) [1] which is at the moment using a PROOF cluster of 13x8-core machines (104 workers). The data used comes from the ongoing ALICE physics data challenges.

- Pilot analysis with local PROOF clusters for fast feedback. This kind of analysis is currently extensively used both on CAF or remote computing centers for training analysis code and debugging the algorithms. This is foreseen to continue at lower scale also after data taking.

- Batch Analysis on the GRID infrastructure. This is the main run mode for accumulating statistics and will be using an important fraction of the available computing resources.

In a context where the full data set is spread over a very large number of resources, parallel data access by several users is a demanding exercise. It makes sense in such conditions to schedule at least the official analysis tasks that implement the physics program of the collaboration. One needs to maximize the profit of having data in memory and optimize the CPU/IO ratio for all different use-cases.

Formalizing the event structure is an important step that was made in this process, but what was even more important was formalizing data access patterns and analysis flow. This has certainly benefits in making best use of the existing resources, but also helps to develop a common and well-tested framework for analysis. The analysis framework implements a common knowledge base and terminology, helps documenting the analysis procedure and makes results reproducible.

3.2 Functionality

ALICE analysis framework allows transparent access to all resources using the same user code, which can be run in parallel environments (PROOF/GRID) in the same way as locally. The access to input data is also transparent to the data type (ESD, AOD or MC truth). An important feature is allowing sharing CPU and memory resources for multiple analysis modules in the same session, using what is called *analysis trains*. The model provides a formalism for defining generic, inter-connectible *analysis tasks* that share a single event loop. User analysis must derive from the generic task class and implement the analysis algorithm.

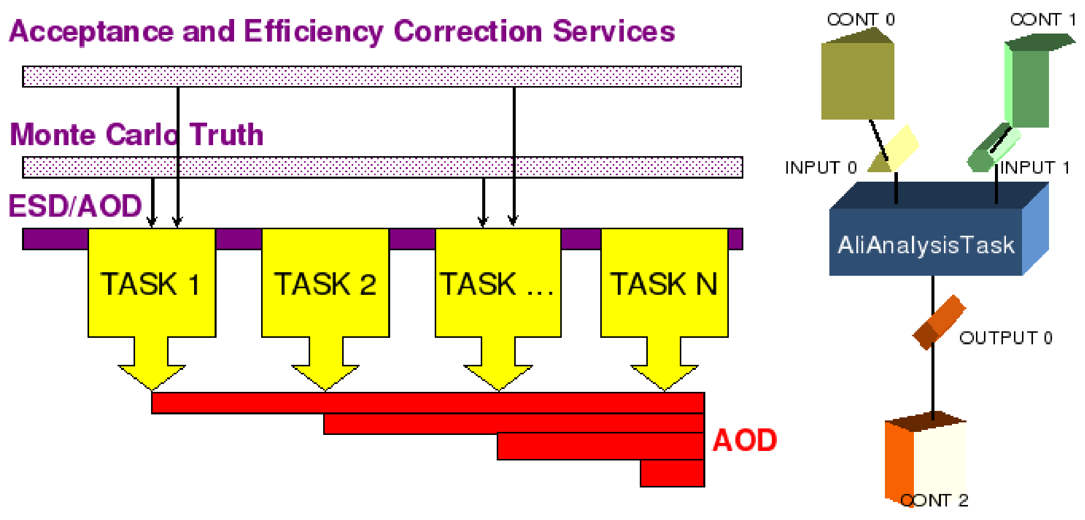


Figure 3: Analysis train based on tasks

As presented in [Figure 3], any task has an arbitrary number of inputs/outputs of declared type that have to be connected to actual data containers. The design is very general and not bound to ALICE software. One task algorithm in this scheme is executed only when actual data becomes available at all inputs. The task has to publish the results for the defined outputs, notifying in this way its readiness for possible clients. Tasks can be connected via containers making a graph and they are all executed event by event by an analysis manager which hides the computing scheme-dependent code from the users.

3.3 Analysis tasks and data management

Each analysis task is executed in three steps:

- Initialization of class data structures (input parameters, output histograms and root trees). Part of this initialization can be done by the user task at creation on the client, while the initialization of the output objects has to be done on the (possibly) remote worker CPU-s. This is formalized as a virtual method to be implemented by the derived task class:

  void CreateOutputObjects()

- Execution of the analysis algorithm. This is done for every event by calling the method:

  void Exec(). This method has to publish the results of the computation at the declared task output slots.

- Finalizing the analysis. Executed after merging all output results via the virtual method:

  void Terminate(). Allows post-processing operations like drawing or fitting histograms.

ALICE event data (MC simulated events, reconstructed ESD events and filtered AOD) derive from a base class that abstracts the common event interface, making possible handling analysis input in a transparent way. This data is accessed from the AliEn (ALICE grid middleware) [2] file catalog via *xrootd*. For interactive parallel processing in CAF, the data files are copied locally on the PROOF cluster and grouped into data sets. In both cases the analysis manager can run the main event loop over a chain of input events. Depending on the event type, additional files may be needed. To make the I/O operations completely transparent to the event type, the framework is using event handlers for accessing and writing the common data (like AOD), as shown in [Figure 4].

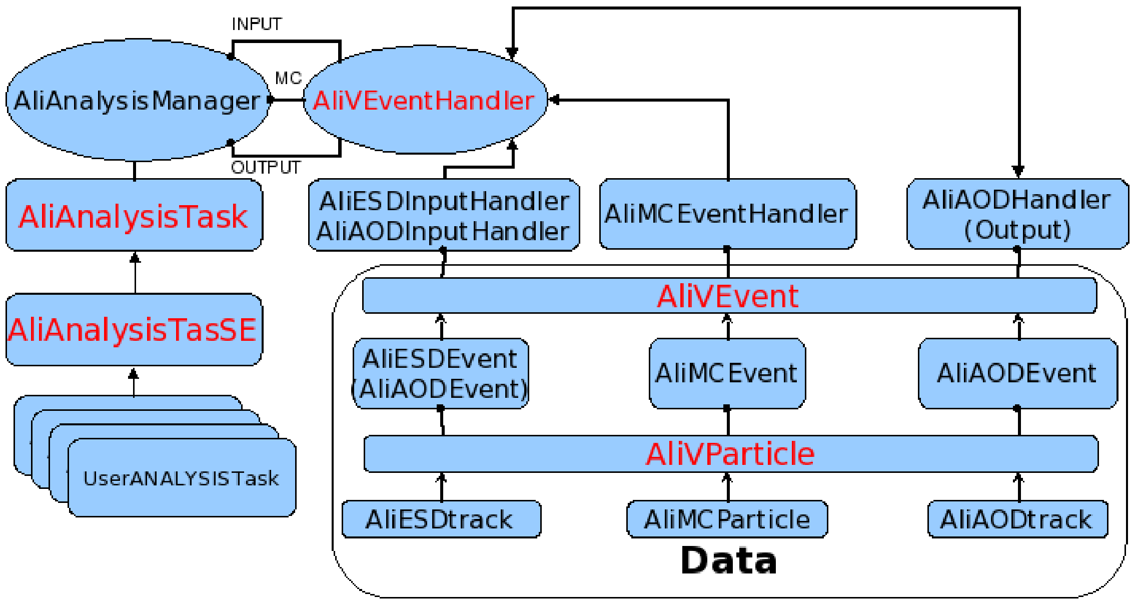


Figure 4: The global picture

3.4 Parallel event processing using PROOF and GRID

PROOF (Parallel ROOT Facility) is using trivial event based parallelism [3] by splitting the input data set into as many chunks as assigned workers in the cluster. The processing time scales very well with the number of workers if the application is not I/O bound and the number of users is constant. In case of the CAF cluster the time for getting back the results when processing 100K pp events is typically in the range of a couple of minutes. PROOF is in addition an interactive system where one can retrieve intermediate results or cancel data processing.

The framework uses PROOF in the following way: the train of analysis tasks is assembled and initialized on the client. The user code can be send to the remote CAF cluster in different ways - via libraries from a custom AliRoot (ALICE offline framework) [4] installation or via PROOF packages (self contained tarballs with the code) which are uploaded and compiled run time. The second option has slower start-up but is very useful in the development stage of analysis algorithms. The analysis manager object created on the client is sent to the PROOF *master* from where it is distributed as workload to the workers together with the chunks of data to be processed. Once on the workers, all tasks book their own output. In case output event handlers are used (like when producing AOD files), the common output tree is also booked before starting the event loop. After all events are processed, the output files are forwarded to a merging service that will send the final result back to the client.

Running the same analysis executed locally in the GRID infrastructure implies few prerequisites: Creation of the input data sets using specific AliEn commands, writing a fully customized job description file and job validation and merging script, copy all dependency files in the AliEn file catalog and keeping their consistency with the used AliRoot version deployed. Once all this is prepared, the analysis can be run using the same code as locally.

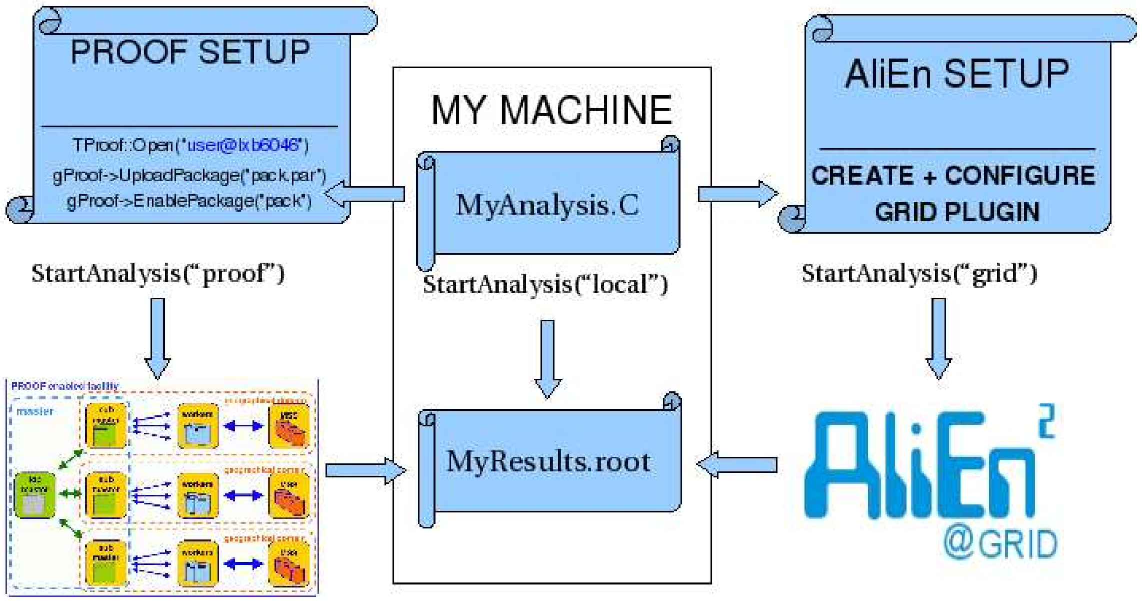

```
TProof::Open("user@lxb6046")
gProof->UploadPackage("pack.par")
gProof->EnablePackage("pack")
```

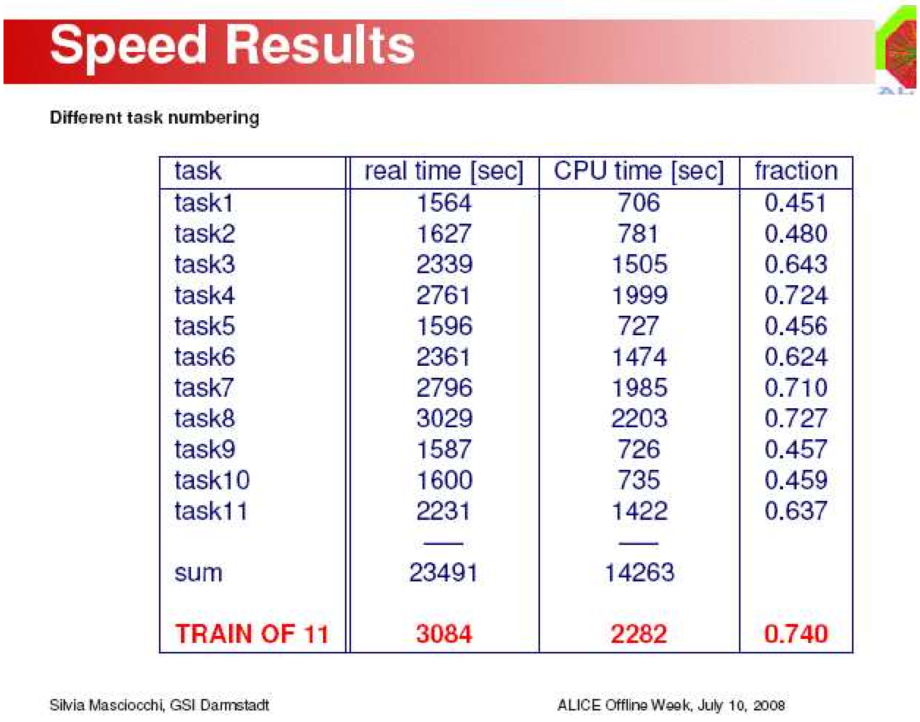
Figure 5: The analysis is transparent to the execution mode up to a simple configuration

A GRID plug-in was developed in order to eliminate (or at least simplify) these prerequisites. This allows the user to operate at ROOT prompt, allowing straightforward customization via API. It automates all interaction with the GRID, keeping the required user actions to a minimum: definition of the requested data path and run numbers, requested software versions, GRID working and output directories. In this way, as summarized in [Figure 5], one can run the analysis code in all different environments and get back the results without needing much specific knowledge of AliEn or PROOF systems.

4. Usage experience and summary

The ALICE analysis framework has been developed and used since more than 2 years. All along this time there was a constant increase of usage of the framework and of the accumulated experience. The migration of analysis code towards *AliAnalysisTask* is now almost completed and currently there are about 15 analysis tasks processing data in the same analysis train. As it can be seen in [5], an *official* analysis train is being tested regularly in all run modes for simulated data. Currently the analysis train is being re-organized while moving from testing/development to the production phase. The experience so far has shown that ALICE CAF was extremely valuable for getting a fast understanding of data as compared to the batch mode. Extensive usage of this analysis facility not only tested the analysis framework, but also pushed it to its limits and therefore triggered new developments in PROOF software. Most of the efforts were done lately for improving the stability of the framework in GRID mode and for hiding its complexity - the new plug-in developed for this purpose will make testing and running analysis trains much easier in batch mode.

User experience has shown that the analysis framework makes much easier to process in an efficient way large data samples. Some tests were done independent from the central development team to check if processing in parallel a train of real analysis tasks is indeed more efficient than running them sequentially. The results presented in [Figure 6] show that this is indeed the case.



Speed Results

Different task numbering

task	real time [sec]	CPU time [sec]	fraction
task1	1564	706	0.451
task2	1627	781	0.480
task3	2339	1505	0.643
task4	2761	1999	0.724
task5	1596	727	0.456
task6	2361	1474	0.624
task7	2796	1985	0.710
task8	3029	2203	0.727
task9	1587	726	0.457
task10	1600	735	0.459
task11	2231	1422	0.637
	—	—	
sum	23491	14263	
TRAIN OF 11	3084	2282	0.740

Silvia Masciocchi, GSI Darmstadt ALICE Offline Week, July 10, 2008

Figure 6: Timing for several analysis tasks submitted separately versus a single train via the framework. Batch cluster at GSI, no GRID latencies.

To summarize, ALICE offline group has developed an analysis framework that hides the external dependencies from the user. The same analysis code that is run on the local machine can be transparently used in PROOF clusters or in batch mode in AliEn grid. The framework has a general design based on independent analysis tasks that can be run in the same ROOT process and can communicate via data. Common data is abstracted at event level and managed by event handlers, while analysis tasks have complete freedom in choosing the algorithm and their private output data. The framework is currently adopted by ALICE collaboration and the experience so far is very positive. Most ongoing development efforts are directed towards framework stability in all use cases, reproducibility and traceability of results.

References

[1] J.F. Grosse-Oetringhaus, *The CERN Analysis Facility - A PROOF Cluster for Day-One Physics Analysis*, *JPCS* **119** (2008) 072017

[2] S. Bagnasco, L. Betev, P. Buncic, F. Carminati, C. Cirstoiu, C .Grigoras, A. Hayrapetyan, A. Harutyunyan, A.J. Peters, P. Saiz, *AliEn: ALICE Environment on the GRID*, Proc. Conf. for Computing in High-Energy and Nuclear Physics (*CHEP*), Victoria, Canada, 2-9 Sep 2007

[3] M. Ballintijn, G. Roland, R. Brun and F. Rademakers, *The PROOF distributed parallel analysis framework based on ROOT*, Proc. Conf. for Computing in High-Energy and Nuclear Physics (*CHEP*), La Jolla, California, 24-28 Mar 2003

[4] F. Carminati, P. Buncic, P. Hristov, A. Morsch, F. Rademakers, K. Safarik, *The AliRoot framework, status and perspectives*, Proc. Conf. for Computing in High-Energy and Nuclear Physics (*CHEP*), Interlaken, Switzerland 27 Sep - 1 Oct 2004

[5] http://aliceinfo.cern.ch/Offline/Activities/Analysis/AnalysisFramework/index.html